International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 5 Issue 4, May-June 2021 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Testing the Component Based Adoption Techniques during Runtime Configuration

Dr. R. Saradha[1], Dr. X. Mary Jesintha[2]

[1]Assistant Professor, Department of Computer Applications, SDNB Vaishnav College for Women, Chennai, India
[2]Teaching Assistant, Department of Computer Science, Salem, Tamil Nadu, India

ABSTRACT

Component based Software Engineering is the most common term nowadays in the field of software development. The CBSE approach is actually based on the principle of 'Select and Use' rather than 'Design and Test' as in traditional software development methods. Since this trend of using and 'reusing' components is in its developing stage, there are many advantages and problems as well that occur while use of components. Here is presented a series of papers that cover various important and integral issues in the field concerned. This paper is an introductory research on the approaches, development process, various phases and assessment available in commercialized models in CBD.

KEYWORDS: *Component Based Design, Design CBD, CBD Approaches, dynamic adoption of CBD*

How to cite this paper: Dr. R. Saradha | Dr. X. Mary Jesintha "Testing the Component Based Adoption Techniques during Runtime Configuration" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-5 | Issue-4, June 2021, pp.496-500, URL: www.ijtsrd.com/papers/ijtsrd42334.pdf

IJTSRD42334

1. INTRODUCTION

For a productive development a technology solely isn't enough, any slightly a lot of complicated project needs management of various aspects that ar on the far side a technology. samples of such aspects ar project coming up with, coordination between project stakeholders, management of resources, organization of labor, and similar. in an exceedingly product life-cycle (i.e. all phases in an exceedingly product's life) technologies ar enablers to specific technical solutions, however additionally catalysts for various development processes. These processes could also be results of specific business or market necessities. so, this can be true within the case of component-based development. Business and market necessities ar drivers of component-based approach. Component-based technologies alter distributed development, parallel development, separation of the event method, increase reusability, etc., that are solutions to the strain on short time-to-market, lower prices or augmented flexibility. There exist several models for software system (and systems) development processes and life-cycles. Most of them are such considering some specific (often non-technical) goals, like quality, foregone conclusion, reliableness, or flexibility, and are typically freelance of technology. samples of such models are completely different sequent models like falls or V model, or reiterative modules like spiral model, or completely different agile strategies, or normal and de-facto standards like ISO 9000, or CMMI. These models ar sometimes laid out in general terms and that they need changes for specific comes.

Some development processes and life-cycle models have their origins in an exceedingly technology or in an exceedingly specific approach. a awfully characteristic example is Object-Oriented Development (OOD) that emprises each technologies and processes. RUP (Rational Unified Process) encompasses a clear influence of OOD. Component-based software system engineering, as a young discipline continues to be centered on technology issues: modeling, system specifications and style, and implementation. there's no established component-based development method. nevertheless, several principles of CBD have vital influence on the event and maintenance method and need respectable modifications of normal development processes.

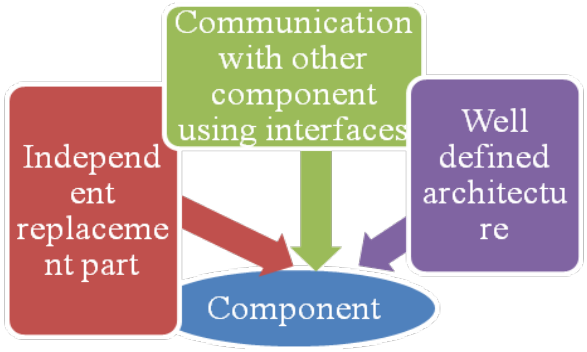

Figure 1: Components and its features

The benefits of part based mostly development embrace lesser development time, lower prices, reusability and

higher modification. A part is that the basic building block of AN application or system created with CBD. Generally, a part are often outlined as AN freelance and interchangeable a part of a system that fulfills a transparent perform. It works within the context of a well outlined design and may communicate with alternative parts through its interfaces Figure one. though the fundamental principle of 'Plug and play is extremely promising, however it additionally brings in some sensible difficulties long-faced by the stakeholders concerned. for example, once we purchase a part, we have a tendency to don't grasp precisely regarding its maintenance, the protection arrangements and also the most vital its behavior once integrated with alternative parts. There exist some models within the market that, to an extent, give North American nation with some standards and interfaces to help the communication method of parts at intervals integration. The models alter the severally designed parts to be deployed and ease the communication between them justifiably explicit, it are often aforesaid that a part model supports parts by forcing them to evolve to sure standards and permits instances of those parts to collaborate with alternative parts during this model. within the absence of part models, there would be obvious non-cooperation among severally developed parts, that the aim of freelance readying and assembled integration of parts.

2. Component-based approach

The main plan of the component-based approach is building systems from already existing parts. This assumption has many consequences for the system lifecycle;

- Separation of the event method. the event methods of part based mostly systems area unit separated from development processes of the parts; the components ought to have already got been developed and presumably are utilized in alternative merchandise once the system development process starts.
- a brand-new process: part Assessment. A new, presumably separated method, finding and evaluating the parts can seem. part assessment (finding and evaluation) are often a neighborhood of the most method, however several blessings area unit gained if the method is performed on an individual basis and also the results of the method could be a repository of parts that has components' specifications, descriptions, documented tests, and also the workable parts themselves.
- Changes within the activities within the development processes. The activities within the component-based development methods are completely different from the activities in non-component-based approach; for the system-level process the stress are on finding the correct parts and corroboratory them, and for the component-level method, style for reprocess are the most concern. for example, the specifics of the component-based development processes we have a tendency to shall use the water model - the only one – however the illustration are often comparatively merely be applied for alternative development processes.

3. Component-based system development process

The main objective of the component-based system development process is the construction of system from (existing) components. This basic characteristic has impact on all phases of the development.

3.1. Requirements Phase

The main objective of the component-based system development method is the construction of system from (existing) elements. This basic characteristic has impact on all phases of the event. In this section the necessities square measure collected, elicited, analyzed, and mere. Associate in Nursing exceedingly in a very} non-component-based approach a necessities specification is an input for the development of the system. During a component-based approach this is often somewhat different; the necessities' specification also will of accessibility of the present elements. This approach will be compared with getting a suit by order from a tailor WHO can create the suit in line with our would like, or by shopping for a suit from a store. Within the second case we have a tendency to couldn't get any suit we have a tendency to like, however take one offered that suits most to our desires. Within the same approach the necessities ought to correlate to the assortment of the elements, i.e., the necessities' specification isn't solely input to the more development, however additionally results of the look and implementation selections. Additional details concerning component-based necessities you'll realize in chapter (Requirements management). 3.2 Analysis & style section the style section of component-based systems follows an equivalent pattern as a style section of software package in general; it starts with a system analysis and a abstract style providing the system overall design and continues with the elaborate design. From the system design, the bailiwick elements are going to be known. These elements don't seem to be necessary an equivalent because the implementation elements. However, they ought to be known and per a close style as assemblies of the present elements. Again, as within the necessities process a exchange between desired style, and an attainable style exploitation the present elements should be analyzed. Additionally, to the present, there'll be several assumptions that have to be taken into consideration: as an example, it should be determined that part model(s) are going to be used, which can have impact on the bailiwick framework additionally as on binding system quality properties.

3.2. Implementation Phase

The implementation activities solely part incorporates committal to writing – truly the additional pure component-based approach is achieved, the less committal to writing is going to be gift. the most stress is placed on element choice and its integration into the system. This method will but need further efforts. 1st the choice method ought to make sure that applicable parts are elite with relation to their useful and extra-functional properties. this might need verification of the element specification, or testing of a number of the component's properties that ar necessary however not documented. Second, it's a acknowledge truth [Wal02] that notwithstanding isolated parts perform correct, an assembly of them could fail, because of invisible dependencies and relationships between them, like shared knowledge shared resources.

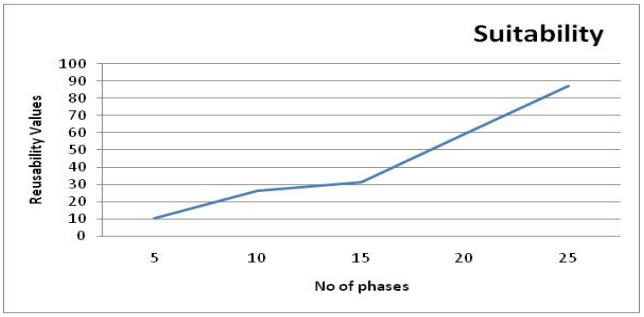

Figure 2: Suitability

This requires that parts integrated in assemblies square measure tested before integrated into the system. the difference of parts could also be needed to avoid fine arts mismatches (such as incompatible interfaces), or to make sure specific properties of the parts or the system. There square measure many proverbial adaptation techniques:

- Parameterized Interface. Parameterized interface makes it potential to vary the element properties by specifying parameters that square measure the components of the element interface. These parameters are often utilized in totally different phases of the element lifecycle, counting on the element model – it are often a building parameter, or a readying parameter or associate execution parameter. associate example of such parameter may be a memory allocation, or frequency of execution, or variety of input file to be received during a row, or similar.
- Wrapper. A wrapper may be a special form of a glue-code that encapsulates a element and provides a brand new interface that either restricts or extends the initial interface, or to feature or guarantee specific properties.
- Adapter. associate adapter may be a glue code that modifies ('adapts') the element interface to create it compatible with the interface of another element. The intention of associate adapter isn't to cover or modify the element properties, however to regulate the interfaces.

3.3. Integration Phase

In a non-component-based development method the combination part includes activities that build the systems from the incoming elements.

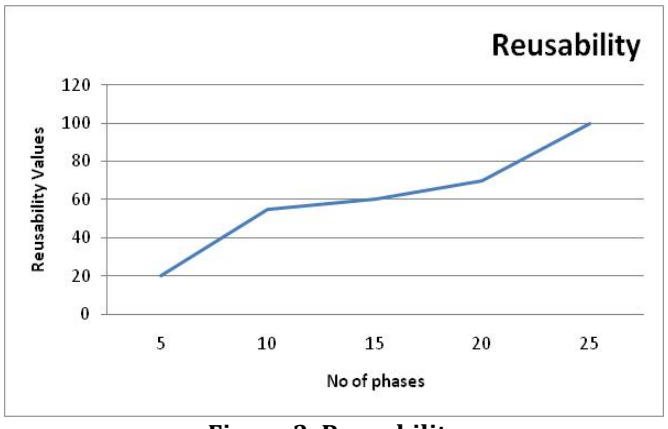

Figure 3: Reusability

The integration part doesn't embody "creative" activities within the sense of making new functions by production of recent code, and for this reason there's demand to automatize and rationalize the method the maximum amount as doable.

The part is but vital because it is the "moment of truth"; several issues come into sight because of beaux arts mismatches of the incoming elements, or because of unwanted behavior of various extra-functional properties on the system level. that's why the combination part is tightly connected to the system take a look at introduce that the system functions and extra-functional properties (in specific rising properties, i.e. properties that don't seem to be visible on element level, however exist on the system level), remains complicated and in several cases as tough as for non-component-based systems. Since system functions don't seem to be completely accomplished by the elements alone, however, usually by a collection of elements to verify these functions the elements should be integrated before the complete system is constructed. For this reason the combination part for component based systems development method is spreading to earlier phases: implementation, style and even within the necessities part. fortuitously in most element-based technologies the component integration is supported by tools, that makes the combination method easier and a lot of economical.

3.4. Test Phase

During the take a look at part the system is being verified against the system specification (including each practical and extra-functional properties). within the water model to take a look at is performed when the system integrations, however, this observes has exhibited several disadvantages. A lot of realistic is changed water model during which the take a look at is performed for code units (such a variant is termed V model). In CBD a requirement for element verification is obvious since the system developers don't necessarily have a bearing on the element quality, element functions, etc., because the element may are developed in another project with alternative functions.

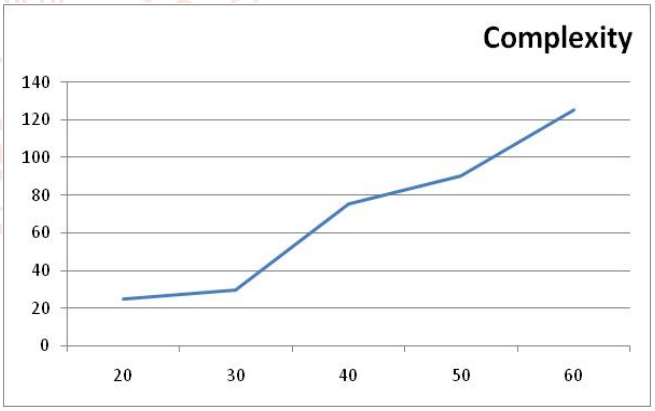

Figure 4: Complexity

The tests performed in isolated components are usually not enough since their behaviour can be different in the assemblies than performing in another environment. The component test can actually be performed many times – by assessment, when integrated in an assembly that provides a particular function, and when deployed (integrated) into the systems.

3.5. Release Phase

The release section includes packaging of the software system in forms appropriate for delivery and installation. The CBD unleash section isn't considerably totally different from a "classical" integration.

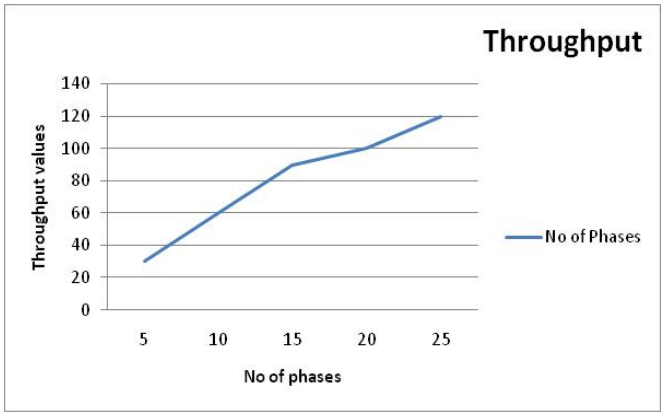

Figure 5: Throughput

3.6. Maintenance Phase

In lifestyle one in every of the patterns of product maintenance is: Repair the merchandise support by replacement bad part. the target of component-based approach for computer code is similar: A system ought to be maintained by replacement of parts. The characteristics of physical (hardware) parts is but totally different from computer code parts. whereas hardware parts will be exposed to a method of degradation in practicality and quality, computer code parts don't amendment. in essence there ought to be no would like for his or her amendment. but the expertise shows the opposite: The accepted law says: "The entropy of a system will increase with time unless specific work is dead to take care of or cut back it." I.e., the computer code can degrade if not maintained. the rationale isn't the degradation of the computer code itself however thanks to the changes of the atmosphere the system runs in. notwithstanding the system functions properly, as time goes it's to be maintained.

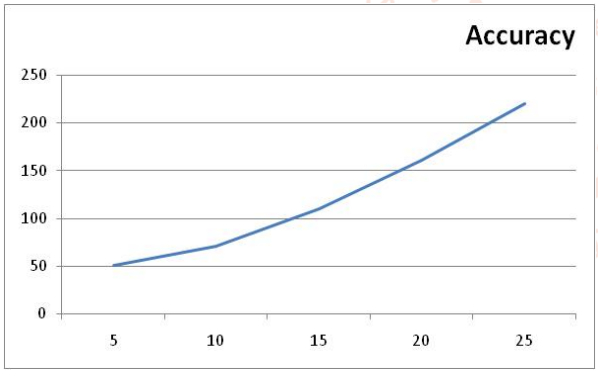

Figure 6: Accuracy

The approach to CBD is to supply maintenance by commutation previous parts by new parts or by adding new parts into the systems. The paradigm of the upkeep method is analogous to the current for the development: notice a correct part, test it, adopt it if necessary, and integrate it into the system

4. Component assessment

Part assessment While development of component-based systems considerably decreases the elaborated style and implementation efforts throughout the system development, it needs extra efforts in alternative activities. As an example rather than implementing needed functions, the developers need to notice parts that offer such practicality. Any they have to verify that chosen parts, i) so offer the required (or virtually desired) practicality, and ii) that the parts will with success be integrated with alternative parts. The consequence is that not the most effective parts (i.e. parts that offer the "best functions") is chosen, however, the parts that work along. To create the system development method economical (i.e., to realize higher time-tomarket) several assessment activities is performed severally and one by one from the system development. A generic assessments method includes the subsequent activities:

- realize – From AN "infinite" element area realize the parts which may offer the desired practicality. This practicality may be a neighborhood of the system being developed, or of a system (or systems) commit to be developed.
- choose – choose may be a refinement of the finding procedure. Between the parts candidates found, choose a element that's most fitted for given necessities and constraints.
- Verify – Inevitable a part of the element choice is that the element verification. the primary level of verification includes testing practical and bound extra-functional properties of a element in isolation. A second develop of verification includes testing the element together with alternative parts integrated in an assembly.
- Store — once an element is assumed to be an honest candidate for the present and/or future applications, it ought to be hold on in an exceeding element repository. The repository can embody not solely the element itself, however additionally extra specification (metadata) that may be helpful in additional exploitation of the element. Example of such knowledge is measured results of element performance, far-famed issues, latency, the tests, and tests results.

Conclusion

In a frame of a selected method model, however similar principles are valid for the other development processes. the most characteristic of part-base development method may be a separation (and parallelization) of system development from component development. This separation encompasses a consequence on alternative activities: Programming problems (low-level style, coding) are less emphasized, whereas verification processes and infrastructural management needs considerably a lot of efforts. This paper analysis the part assessment, Component-based system development method, part primarily based approaches, demand, Design, Implementation, Integration, check and maintenance method levels phases.

REFERENCES

[1] H. Hansson, M. Åkerholm, I. Crnkovic, M. Törngren, "a Component Model for Safety-Critical Real-Time Systems", in Proceedings of 2004 30th EUROMICRO Conference (EUROMICRO'04), France.

[2] "Research Areas of the Software Engineering Group", Online Available:http://www.cs.uni-paderborn.de/en/research-group/software-Engineering/research/research-areas.html

[3] "Basic Concepts of Component-based Software", Online http://www.idt.mdh.se/kurser/cdt501/2008/lectures/book%20Basic%20Concepts%20of%20CBSE.pdf

[4] Murat güneştaş , "A study on component based software engineering", a Master's thesis in Computer Engineering, Atılım University, JANUARY 2005

[5] Ivica Crnkovic; Stig Larsson; Michel Chaudron, "Component-based Development Process and Component Lifecycle." Online Available: http://www.mrtc.mdh.se/publi cations/0953.pdf

[6] Luiz Fernando Capretz, "Y: A New Component-based software life cycle model", Journal of Computer Science 1 (1): 76-82, 2005, ISSN 1549-3636 © Science Publications, 2005.

[7] K. Kaur; H Singh, "Candidate process models for component based software development", Journal of Software Engineering 4 (1):16-29, Academic Journal Inc, India 2010.

[8] Syed Ahsan Fahmi; Ho-Jin Choi, "Life Cycles for Component-Based Software Development", IEEE 8th International Conference on Computer and Information Technology Workshops 2008.

[9] A. W. Brown, K. C. Wallnau, —The Current State of CBSE,||IEEE Software, Volume: 15 , Sept.-Oct. 1998, pp. 37- 46

[10] C. Szyperski, "Component Software: Beyond Object-Oriented Programming," Addison-Wesley, New York, 1998.

[11] G. Pour, —Enterprise JavaBeans, JavaBeans & XML Expanding the Possibilities for Web-Based Enterprise Application Development,|| Proceedings Technology of Object-Oriented Languages and Systems, 1999, TOOLS 31, pp.282-291.

[12] Component-based Development Process and Component Lifecycle by Ivica Crnkovic, Stig Larsson, Michel Chaudron

[13] Component Model with Support of Mobile Architectures' by Marek Rychllli, Brno University of Tcchnology, Czech Republic

[14] The Koala Component Model for Consumer Electronics Software, Rob van Ommering, Frank van der Linden, Jeff Kramer, Jeff Magee, IEEE Computer, March 2000, p78-85